Indiana University-Purdue University Fort Wayne

Design Report

CPET 491-Senior Design Phase II~~2~~

A Web-Enabled Parking Garage Monitoring System for Real-Time Data & Trend Analysis

Jacob Pitcher GRADE A, and~~,~~ Andrew White GARDE: A

Excellent job!

October 1, 2009

Course Professor:
Paul I-Hai Lin, Professor of Electrical and Computer Engineering Technology

Technical Advisor:
Gary Steffen, Associate Professor and Chair of Electrical and Computer Engineering Technology

This project consists of a mechanism of which the primary purpose is to count vehicles entering and leaving a parking garage, and to display that count information in several

ways to the end user.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF EQUATIONS

LIST OF TABLES

I. Introduction

A. Problem Statement

Indiana University-Purdue University Fort Wayne (IPFW) is rapidly growing, as evidenced by an increase in enrollment of 11% in the Fall semester of 2009 [1]. Out of 2317 active Spring semester 2009 students surveyed, approximately 48% have shown dissatisfaction with parking availability on campus, 68% report having experienced a traffic jam in a parking garage on campus, and 53% of students would avoid parking in a garage when capacity is displayed as near-full [2]. Currently, the university has two parking garages with little notification on their capacity. The parking garage closest to the Engineering, Technology, and Computer Science building is equipped with two LED displays which are capable of displaying dynamically updated information. Currently, Police and Safety places a static message on the parking garage when the number of cars nears the garage capacity. A more accurate system would benefit both students and administrators in savings of both time and frustration.

B. Solution Statement

The overall goal of this project is to address the need for notification of parking availability in the parking garages on campus. Notification must be available to all students and employees of the university who make use of said parking garages. Notification must be performed both on a chance basis at the time the user requires it and on user demand at any given time. Both chance notification and on-demand notification will be accomplished through the use of garage-mounted displays, a web server, and optionally a Voice Over IP (VOIP) or Short Message Service (SMS) server as shown in ~~(~~See Figure 1~~Figure 1Figure 1)~~.

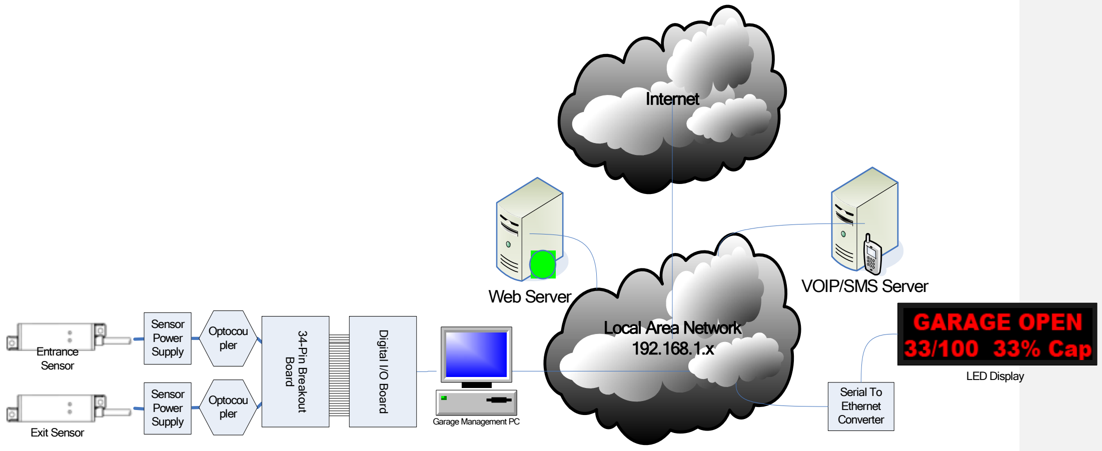


Figure 1 – Hardware System Architecture

Expenses for this project must be minimized through acquisition of donated parts from as many vendors as possible. Any major expenses must be paid for through acquisition of funding from campus organizations. This project, which must lay the groundwork for a fully-implementable system, must be completed within the Fall Semester of 2009, with future expansion provided by students who choose to continue with implementation.

C. Objectives

Primary objectives of this project include designing, building, and testing of a prototype car counting system within the time period of one semester without running out of time and without excessive cost to the project designers. The car counting system should minimize the effort a student must expend in order to determine the number of cars in a parking garage at any given time. The system should be implemented in such a way that the speed, volume, or size of vehicles entering a garage at any time does not significantly impact the reliability of the system. Conveyance of the garage capacity and utilization should be performed in such a way that a student may easily decide which garage to enter, or to attempt entry at all.

II. Discussion

A. Hardware Subsystems

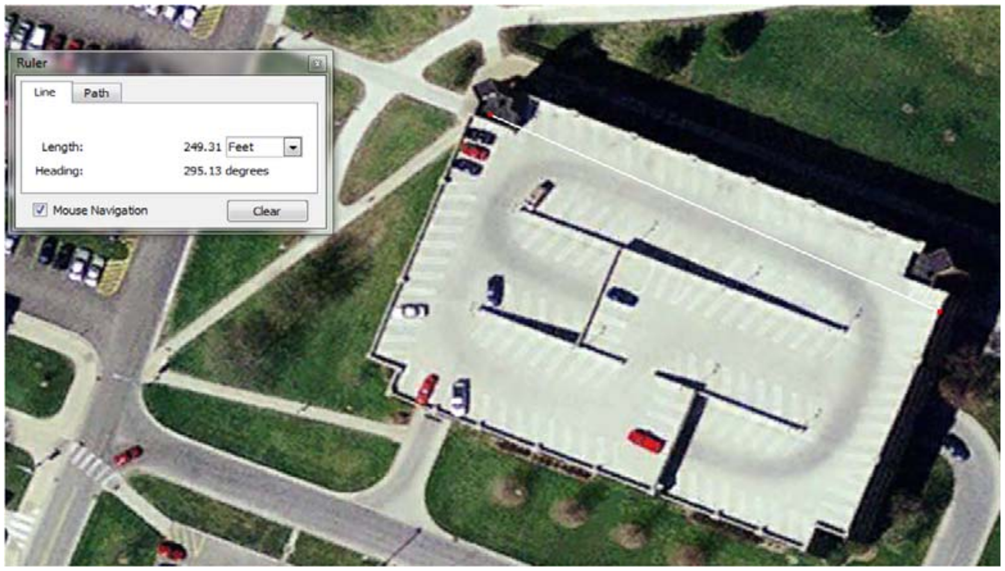
Major computer hardware subsystems utilized in this project include a Sensor Server, a Web Server, and, optionally, a Voice Over IP / Short Message Service server. Emphasis shall be placed on the accurate and expedient communication of count information to the end user. Communication shall be performed through several user-selectable channels. Channels shall consist of visual and/or auditory notification of the parking garage status.

1. Sensor Server

The sensor server shall consist of an Intel x86 (32-bit) architecture computer running a Windows operating system. The 32-bit architecture is a requirement of the Digital I/O board ~~we~~ chosen ~~to use~~ for this project. Our choice of the USB-I2C/IO board was based on the simple I/O connection (34-pin ribbon cable similar to a floppy drive cable), abundant availability of I/O ports (32 user-configurable bits), easy power and connectivity requirements (powered by and connected with a USB port), and availability of the vendor for provision of sample components [3]. The choice of operating system is governed by the library interface provided by the Digital I/O board vendor. The web server itself shall be supplied with at least 512MB of memory, a 32GB hard drive, and at least a 1.4GHz processor, based upon the minimum system requirements of Windows Server 2008 Standard Edition [4].

a) Sensor Requirements and Specifications

The sensors used in this project shall consist of Banner Engineering Q7MB Flat-Pak Vehicle Detection Sensors. Each sensor shall be placed either in the center of a traffic lane or at the side of a traffic lane. Placement concerns include accidental detection of other lanes of traffic due to sensitivity levels, and distance to the garage controller unit (There is a maximum length of 60m (200 feet) on a cable run [5]. According to measurements performed on Google Earth, the distance from the furthest (southeast) entrance and exit to the northeast corner of the parking garage is greater than 200 feet, meaning that even under ideal (straight-line) conditions, a wired sensor simply would not reach the distance from the entry/exit point to the garage controller unit (See Figure 2~~Figure 2Figure 2~~).



Length: 249.31 Feet
Heading: 295.13 degrees

Figure 2 – Engineering Technology Parking Garage Measurement

As such, the sensor sensitivity and exact location must be determined on-site as conditions may vary, and an actual implementation shall be based upon the wireless version of this sensor technology. Actual implementation of the wireless version of this technology is left as an exercise for future CPET/ECET students, as the study of these components will not significantly contribute to the overall design of the project, and because the input and output to the wireless version of these sensors is the same as it is to the wired version.

b) Sensor Interface

The sensor interface, as shown in Fgure 3, is a system which is composed of the sensor, sensor power supply, input optocoupler, output optocoupler, and digital I/O board (~~See Figure 3~~ avoid using this type of referring to the figures used in the report).

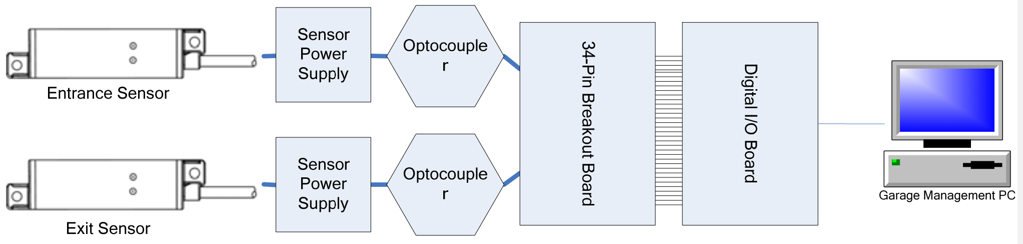


Figure 3 - Sensor System Block Diagram

The sensor itself is a Banner Engineering M-GAGE Q7M Flat-Pak Vehicle Detection Sensor. The sensor requires a voltage in the range of 10 to 30 volts DC, with a maximum voltage fluctuation of 10% at 43mA when the temperature is between 50 degrees Celsius. If the temperature is above 50 degrees Celsius, the supply voltage is between 10 and 24 volts DC, again with 10% maximum voltage fluctuation. ~~(See~~ Table 1~~Table 1Table 1~~ shows M-GAGE Q7m Flat-Park specifications~~)~~.

Table 1 - M-GAGE Q7m Flat-Pak Specifications [5]

Supply Voltage	10 to 30V dc (10% max. ripple) at 43 mA, exclusive of load Above +50° C, supply voltage is 10 to 24V dc (10% max. ripple)
Sensing Range	See Figure 4~~Figure 4Figure 4~~ and Figure 5~~Figure 5Figure 5~~.
Sensing Technology	Passive 3-axis magnetoresistive transducer
Supply Protection Circuitry	Protected against reverse polarity and transient voltages
Output Configuration	Two SPST solid-state outputs conduct when object is sensed; one NPN (current sinking) and one PNP (current sourcing).
Output Protection	Protected against short-circuit conditions
Output Ratings	100 mA maximum (each output) **NPN saturation:** < 200 mV @ 10 mA and < 600 mV @ 100 mA; **OFF-state leakage current:** < 200 microamps **PNP saturation:** < 1.2V @ 10 mA and < 1.6V @ 100 mA; **OFF-state leakage current:** < 5 microamps
Output Response Time	20 milliseconds
Delay at Power-Up	0.5 seconds
Temperature Effect	< 0.5 milligauss/°C
Adjustments	Configuration of Background Condition and Sensitivity Level may be set by pulsing the gray wire remotely via the portable programming box (see page 3).
Indicators	**Two Indicators** (see Figure 2 and instructions on page 3):Power Indicator (Green)Configuration/ Output Indicator (Red/Yellow)
Remote TEACH Input	Impedance 12K ohms (low = < 2V dc)
Construction	**Housing:** Anodized aluminum **End Caps:** Thermoplastic polyester
Operating Conditions	-40° to +70°C (-40° to +158° F); 100% max. rel. humidity
Connections	Shielded 5-conductor (with drain) polyethylene jacketed attached cable or 5-pin Euro-style quick-disconnect PVC pigtail (see page 8 for quick-disconnect cable options)
Environmental Rating	Leak proof design is rated IEC IP69K; NEMA 6P
Vibration and Mechanical Shock	All models meet Mil. Std. 202F requirements method 201A (vibration: 10 to 60 Hz

Formatted: Font: (Default) Arial, 9 pt

Formatted: Font: 9 pt

Formatted: Font: (Default) Arial, 9 pt

Formatted: Font: 9 pt

	max., double amplitude 0.06", maximum acceleration 10G). Also meets IEC 947-5-2; 30G 11 ms duration, half sine wave.

The sensing mechanism consists of a three-axis magnetoresistive transducer which detects the earth’s magnetic field. When a ferromagnetic object interrupts this field, it is registered as a change relative to the baseline reading taken when the sensor is configured. Internally, the perceived change in magnetic field is called excess gain, and is essentially the “extra” signal detected when a ferromagnetic object passes over or near the sensor. ~~(See~~ Figure 4~~Figure 4Figure 4~~ and Figure 5~~Figure 5Figure 5~~ show ~~for~~ examples of mounted 1 meter above ground and 0.25 meters below ground~~)~~ AND may be rewrite it [5]. Each sensor has two outputs, a PNP (current sourcing) output and an NPN (current sinking) output (See Table 1~~Table 1Table 1~~ for more information).

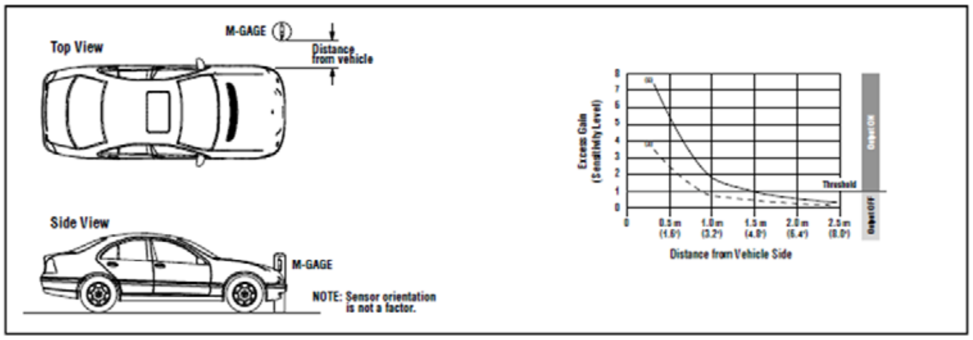


Figure 4 - Sensor Mounted 1 Meter Above Ground [5]

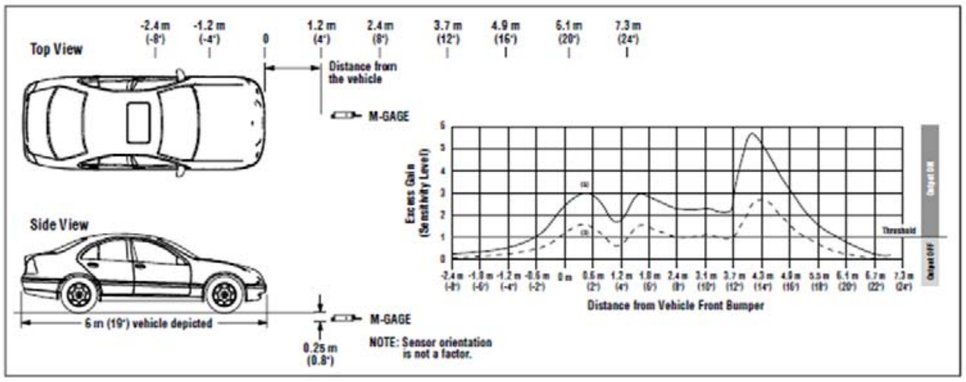


Figure 5 - Sensor Mounted 0.25 Meters Below Ground [5]

c) Power Supplies

Each sensor requires its own power supply. For this project we have chosen to use the Banner Engineering PS115-1 Power Supply which sources up to 100 milliamps, includes the remote teach function required by our application, and is powered by 105-130V AC (See Figure 7~~Figure 7~~~~Figure 7~~?? Reword the sentence) [6]. Internally, power is distributed to these power supplies from a Qualtek EMI filter which provides filtering to the system from line noise, integrates a 6A fuse, and provides a simple connection for a standard PC power cable to power the system [7]. Our implementation involves the ground wire being connected to the chassis of the mounting box, which will be connected to earth ground. ~~See~~ Figure 8~~Figure 8~~~~Figure 8~~ shows ~~for~~ sensor supply voltage and load current requirements.

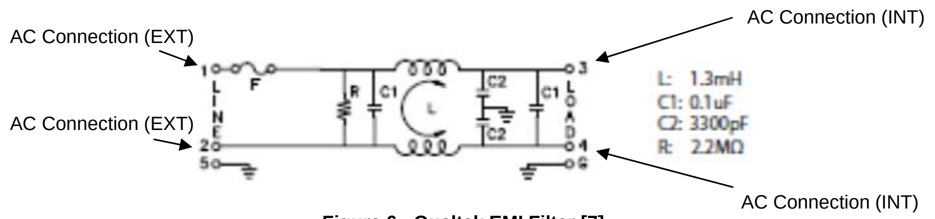


Figure 6 - Qualtek EMI Filter [7]

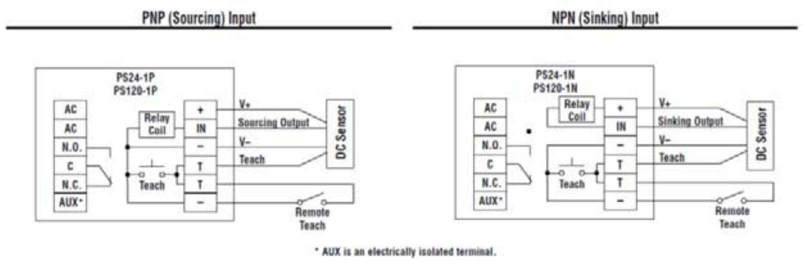


Figure 7 - Banner Engineering PS115-1 Power Supply [6]

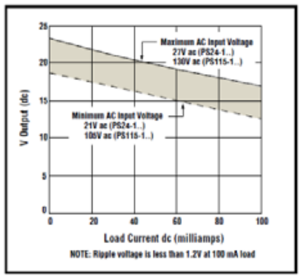


Figure 8 - Voltage Regulation vs. Load [5]

d) Sensor and Sensor Subsystems

Sensor testing has revealed that an above-grade mounted sensor is too unreliable in detecting vehicles, which means that an actual implementation of the project will require below-grade mounted sensors in the center of the lane to accurately count vehicles. Testing was performed with a 2005 Pontiac Grand Prix and the collected data is organized in ~~(~~See Table 2~~Table 2Table 2~~~~)~~. This is further supported by a note from Banner Engineering stating that for optimal performance in detecting vehicles, that the sensor should be mounted below grade in the center of the traffic lane [5]. ??See Figure 4~~Figure 4Figure 4~~ and Figure 5~~Figure 5Figure 5~~ for what info??.

Table 2 - Sensor Testing

Distance Vehicle Detected	Height Above Ground	Distance From Vehicle	Vehicle Speed	Sensitivity Level	Errors?
0	18 inches	9 inches	5mph	6	Tripped Twice
0	18 inches	9 inches	5mph	5	No Trips
0	18 inches	9 inches	5mph	4	No Trips
0	18 inches	9 inches	5mph	3	No Trips
0	18 inches	9 inches	5mph	2	No Trips
0	18 inches	9 inches	5mph	1	No Trips
16 feet	30 inches	9 inches	2mph	6	
0	30 inches	9 inches	2mph	5	No Trips
0	30 inches	9 inches	2mph	4	No Trips
0	30 inches	9 inches	2mph	3	No Trips
0	30 inches	9 inches	2mph	2	No Trips
0	30 inches	9 inches	2mph	1	No Trips
12 feet	30 inches	19 inches	2mph	6	
	30 inches	19 inches	2mph	5	No Trips
	30 inches	19 inches	2mph	4	No Trips
	30 inches	19 inches	2mph	3	No Trips

	30 inches	19 inches	2mph	2	No Trips
	30 inches	19 inches	2mph	1	No Trips

The minimum response time of the sensor is 20 milliseconds, meaning that the minimum amount of time a car must activate the sensor for is 20 milliseconds. Assuming that the distance the vehicle is detected is 16 feet, this means that a car would have to drive by the sensor at 545.45 miles per hour in order to not trip the sensor (See Equation 1~~Equation 1Equation 1~~). This is simply not possible given that the speeds of a Formula One car even at the end of a straight on the Indianapolis Motor Speedway are in excess of 200 miles per hour [8]. A car simply could not reach 545.45 mph given the limited amount of space provided in most parking garages. It should also be noted that at two miles per hour, the sensor will be tripped for approximately 5.5 seconds assuming the vehicle is detected for 16 feet (??See Equation 2~~Equation 2Equation 2~~??). This shows that for a line of vehicles entering the garage, the sensor activation time will be large in proportion to the sensor deactivation time. ??See Figure 9~~Figure 9Figure 9~~?? for a sample of the sensor testing rig which was used to gather the data in Table 2~~Table 2Table 2~~.

Distance Travelled / Minimum Sensor Activation Time = Vehicle Speed To Avoid Detection

Equation 1 - Vehicle Speed Required To Avoid Detection

Distance Travelled / Vehicle Speed = Sensor Activation Time

Equation 2 – Sensor Activation Time At Two Miles Per Hour

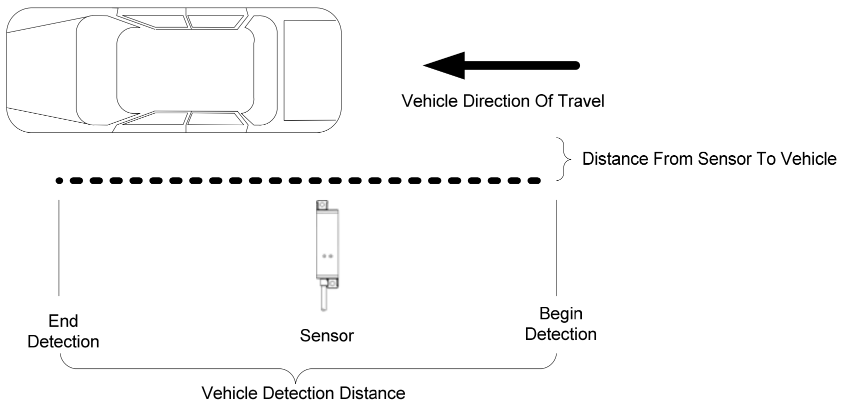

Vehicle Direction Of Travel
Distance From Sensor To Vehicle
End Detection
Sensor
Begin Detection
Vehicle Detection Distance

Figure 9 - Sensor Testing Rig

Each sensor has five wires with various functions including positive and negative power, ground, PNP and NPN outputs, and a teach function (See Table 3~~Table 3Table 3~~). It should be noted that in an actual implementation of the system, we suggest that the connections for the sensors be moved to the outside of the system box, and that they use the standard Banner Engineering Euro-style connectors [5].

Table 3 - Sensor Connections and Functions

Wire Color	**Function**
Brown	Positive Power
Blue	Negative Power
Silver (bare)	Ground
Black	NPN Output
White	PNP Output
Gray	Teach

e) Input Subsystem

The NPN and PNP outputs of the sensor (10-30 volts DC) are connected to pins A1 and A2 of the ABB optocoupler (See Figure 10~~Figure 10Figure 10~~), which essentially provides power to the optocoupler and activates it simultaneously, supplying current to the base of the internal transistor and allowing current to flow from the collector to the emitter, pin 13 to pin 14 on the optocoupler, which effectively pulls the corresponding pin of the digital I/O board to ground (presuming pin 13 is connected to an input port and pin 14 is connected to ground). This effectively transmits a TTL “zero” to the digital I/O board when the sensor is activated (??See Figure 3~~Figure 3Figure 3~~??) [9].

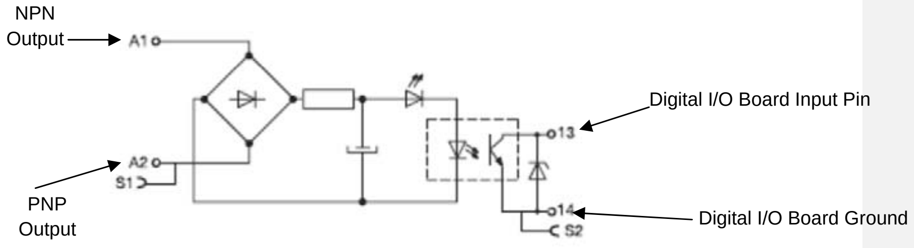


Figure 10 - ABB 1SNA 645 021 R2600 Optocoupler Schematic [9]

f) Output Subsystem

Output from the Garage Monitoring System to the sensor merely consists of the teach function and the sensor sensitivity function. Essentially when a TTL “one” is written to an output port of the digital I/O board, that data is transmitted to pin 1 of the Weidmuller optocoupler (Pin 2 is connected to Ground, and Pin 3 is connected to a 5 volt source to supply voltage for the internal amplifier). This transmission of a TTL “1” to the

Digital I/O Board +5V DC

Remote Teach Pin

Digital I/O Board Ground

Digital I/O Board Output

Sensor Power Supply DC - Pin

optocoupler activates the device and provides a connection between pins 4 and 5 of the optocoupler, which essentially takes the remote teach pin of the power supply and brings it to the level of the DC – pin of the power supply for however long the optocoupler is activated (??See Figure 11~~Figure 11Figure 11~~??) [10].

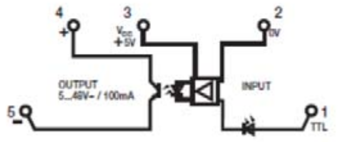


Figure 11 - Weidmuller 839894 Optocoupler Schematic [10]

2. Web Server

The web server shall consist of a 32 or 64-bit computer running either a Windows or Linux operating system. Freedom of choice of operating system and hardware architecture in this case has led us to the decision to run our Web Server on a 32-bit Windows computer due to the ease of availability of such hardware for implementation in the prototype of this system. Web server hardware shall be dependent upon the number of simultaneous user requests, which will be limited in the software. The web server software does not specify a minimum system requirement, however we plan to provide our web server with at least 512MB of memory, a 32GB hard drive, and at least a 1.4GHz processor [4]. The system must have a TCP/IP network interface.

3. VOIP / SMS Server

The VOIP / SMS server is a system which we have not had time to fully investigate, however it has been determined that an email message may be sent to most phones. The system will additionally be running Windows Server 2008, and as such it will have a minimum hardware requirement of 512MB of memory, a 32GB hard drive, and at least a 1.4GHz processor [4]. The system must have a network interface.

B. Software Subsystems

1. Sensor Server Software

The parking garage monitoring software shall consist of a 32 bit windows application designed to constantly monitor and record the input provide by the hardware and also provide the ability for the user to configure the hardware. Main classes (objects) of the program consist of the garage class, the sensor class, various user-interface forms, and the main form which launches the program. (??See Figure 12~~Figure 12Figure 12~~??).

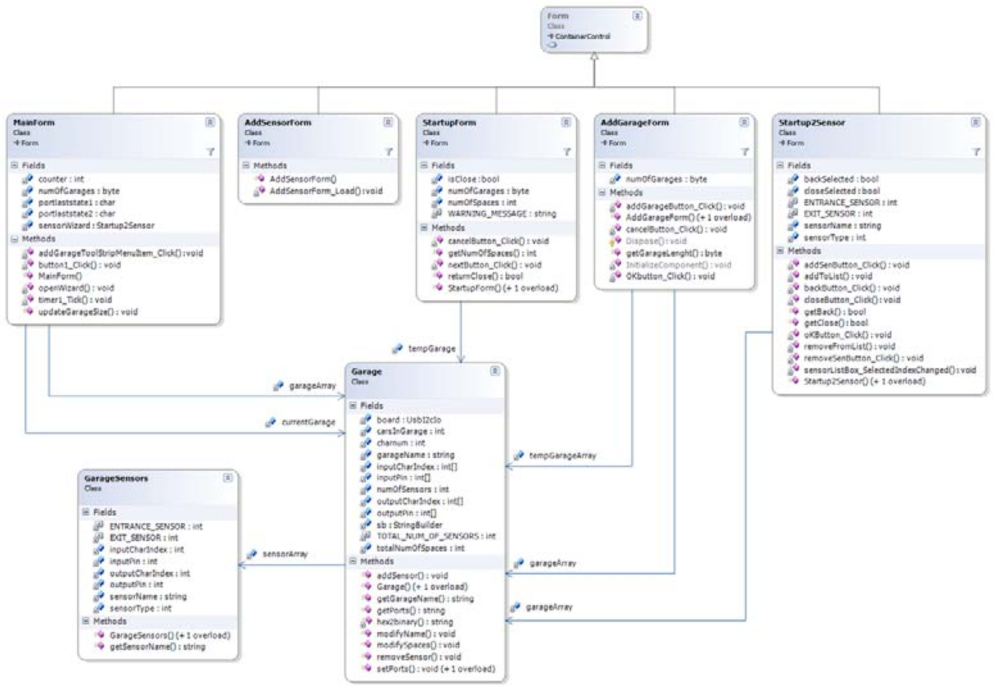


Figure 12 - C# Program UML Diagram

This software will interface with an LED display sign via a Dynamic Link Library (DLL) file which will present the recorded information to the user via the parking garage. The software will have four main states, consisting of the main timer loop in which the program will spend most of its time, the exit sensor state, the entrance sensor state, and the FTP upload state (??See Figure 13~~Figure 13Figure 13~~??). The program begins in the Begin State, proceeds automatically into the Setup state where the user is prompted for garage and sensor information. From that point, the application proceeds to the main timer loop where it spends most of its time. If an entrance or exit sensor is tripped, the application proceeds to the corresponding sensor state and takes appropriate action, which consists of either incrementing or decrementing the counter and writing to the database. Periodically, the application will proceed to the FTP upload state where a text file containing the garage name will be uploaded to a web server for additional processing and user-message delivery on the web server. The application will also provide functionality to perform trend analysis, displaying a line-graph showing the number of cars over a period of time to an administrative user, however this feature is currently under development.

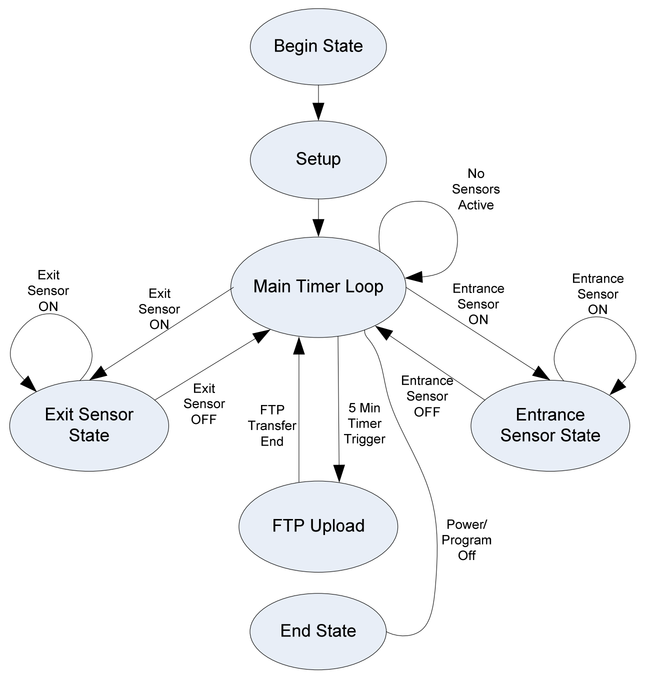


Figure 13 - State Transition Diagram

Detailed requirements and specifications are as follows (??See Figure 14~~Figure~~ 14~~Figure 14~~??):

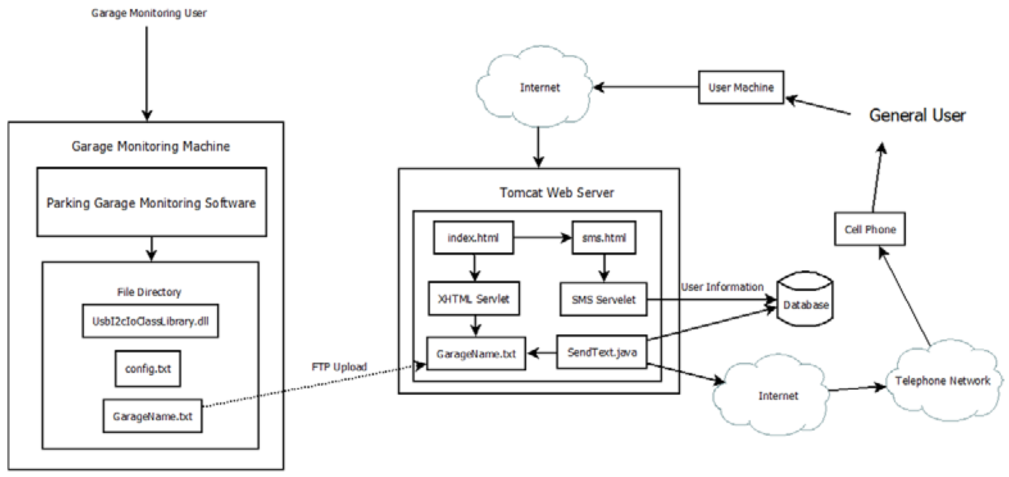


Figure 14 - Software System Architecture

- Monitoring software will be written in the Microsoft Visual Studio C#.NET language.
- Software must implement the UsbI2cIoClassLibrary.dll provided by the digital I/O vendor [3]. This DLL file will provide the communication with the digital I/O board.
- Each garage within the software will be assigned one digital I/O board. Each board will be able to accommodate a maximum of 10 magnetic vehicle detectors.
- Software must instantiate a garage object data type by referring to the board "instance". This instance is determined by the number of boards currently connected to the system. The first instance must be represented by a byte data type of value "0". Each additional instance will be incremented by one byte.
- A maximum of 5 garages can be stored in the program.
- When the software loads, the initialization constructor of the main form will check for a configuration file consisting of the garage name, number and type (entrance, exit) of vehicle sensors, and garage capacity. If the file exists, the program will create a new garage instance, read the file line by line, and store the information in the newly created garage object. If this file does not exist, a form wizard will launch that prompts the user to enter a garage name and the total amount of parking spaces in the garage. Both of these fields must be entered via a textbox.

- The first wizard form must contain a “Cancel” button which will terminate the program. The form must also contain a “Next” button which will allow the user to proceed to the next form of the wizard. (??See Figure 15~~Figure 15Figure 15~~??)

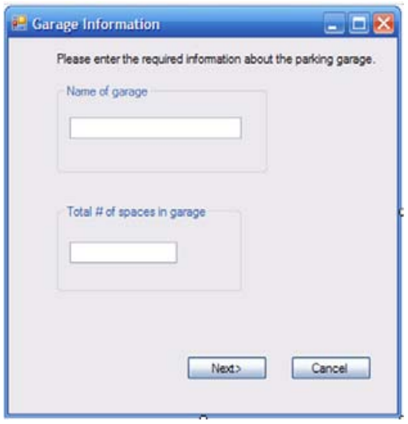

Please enter the required information about the parking garage.

Figure 15 - Garage Information Entry Form

- A second wizard form will prompt the user to optionally add a sensor(s). If the user wishes to add a sensor, he/she must enter the name of the sensor as well as the type (entrance, exit). The sensor name must be entered into a text box and the sensor type must be selected via a radio button. This information will be stored in a garage object containing a maximum of 10 Sensor objects. (??See Figure 16~~Figure 16Figure 16~~??)

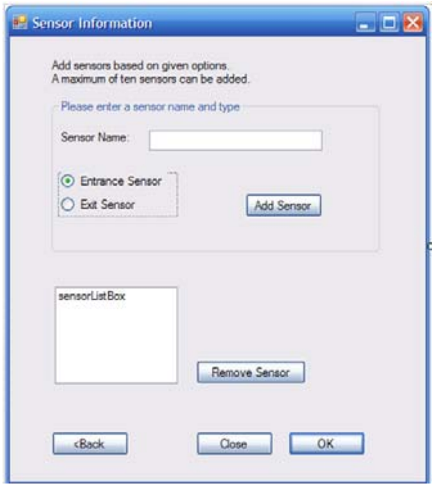

Add sensors based on given options.
A maximum of ten sensors can be added.
Please enter a sensor name and type

Figure 16 - Sensor Information Entry Form

- The second wizard form must contain a “back” button returning the user to the first wizard screen, a “cancel” button which terminates the program, and a “finish” button which closes the wizard.
- Both wizard forms must handle error checking. Text fields that require a user input must not contain an empty string. Text fields that require numerical data must contain a string that can be parsed to an integer data type value.
- All information concerning the garage must be saved to a configuration file located in the program folder.
- During the initialization of the main form, a timer object must be created to poll the vehicle sensors every 50ms.
- During the creation of a sensor object, the garage class must assign a pair of board pins that the sensor will connect to. The pin information will be stored in separate parallel arrays consisting of the following data respectively: physical input pin (int data type), physical output pin (int data type), 32 bit string input character position (int data type), and 32 bit string output character position (int data type).
- Every event triggered by the timer will call a method to poll the input pins of the board.
- The pins of the board are represented as a 32 bit unsigned integer which is parsed string. Each sensor must compare its input character position to the corresponding position of the input character string to determine the state of the sensor’s assigned input pin.
- If the input pin of the sensor is read as a “0”, based on the sensor type, the current car count variable of the garage will either increment or decrement by one integer value.
- A state variable must be set true once an input is detected. This variable will remain true until the state changes.
- Each time a vehicle is detected, the current number of cars in the garage as well as a time stamp of the event will be written to a log file.
- Each time a vehicle is detected, the current number of cars as well as the percentage value (current # of vehicles/ capacity of garage X 100) of cars in the garage will be written to the LED sign via a DLL method.
- This DLL is included in Software Development Kit (SDK) which can be purchased from the manufacture of the LED sign. If funding is not obtained to purchase the SDK, the LED sign will be simulated in the software.
- A sensor configuration form will be accessed from a file menu. This configuration file will be the same form as the wizard form with the exception that the “back” button will be hidden from the user.
- This form must allow the user to add/remove sensors from the garage.

- Every 5 minutes, a second log file will be written. The file's name will be formatted as: GarageName.txt. Each line in the file will contain a timestamp followed by the amount of cars in the garage when the file was written as well as the total amount of parking spaces. This file will be used by the web server.
- The main form will contain the following: Garage name, current number of cars in the garage, total number of spaces in the garage, percentage of spaces taken, a list of every sensor in the garage, and a warning if the garage is near maximum capacity.
- Next to each sensor will be an indicator depicting when the sensor is tripped.
- Each sensor object is its own thread and the bit of the un-signed integer corresponding to the output ort of the digital I/O board which is connected to the teach pin of the corresponding sensor will be pulsed high. During a specified interval, the thread will sleep in order to avoid detecting input pulses on that sensor.

2. Web Server Software

a) HTML/Servlet Web Page

A web page will be provided for the user to display the information of the parking garage. The webpage will display the garage name, the current amount of cars in the garage, and the percentage of cars in the garage. A web link will be provided to a page which will allow the user to subscribe to text alerts (??See Figure 17~~Figure 17Figure 17~~??).

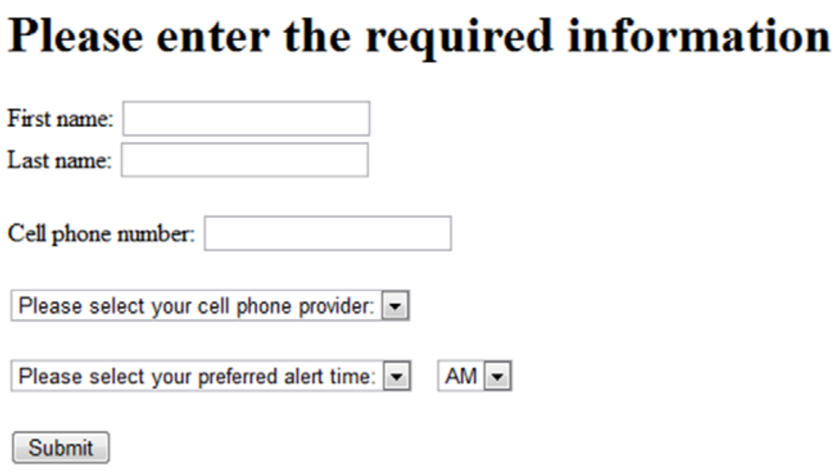

Please enter the required information
First name:
Last name:
Cell phone number:
Please select your cell phone provider:
Please select your preferred alert time:
AM
Submit

Figure 17 - SMS Text Registration Page

We are currently developing the servlet and Java application which will handle this portion of the project, and as such there is no UML diagram yet. The Java application which sends the text alert will poll a database every half hour in order to determine if any users have requested an SMS alert at that time. If a time period arrives during which a user has requested an SMS alert, the application will send an email to the requesting user's phone, and write an entry to a log file specifying that this has occurred. Every week the application will archive the log file so that it does not become too large.

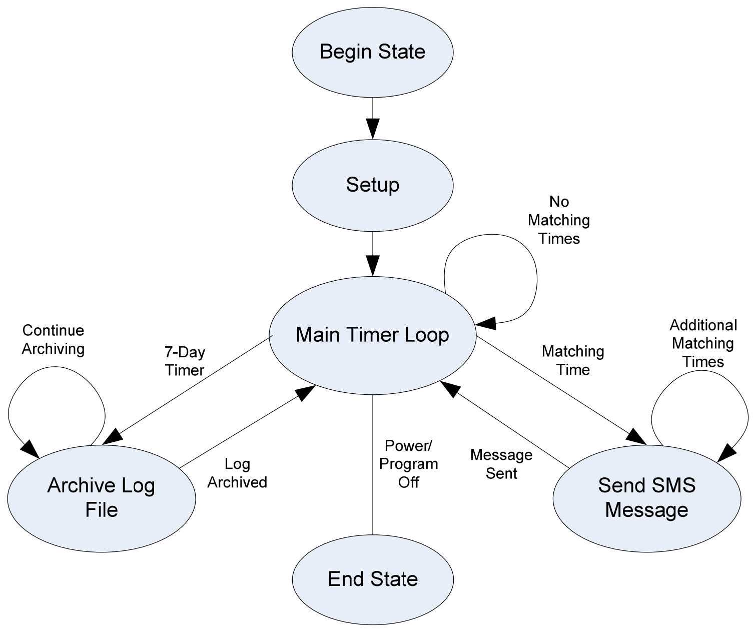


Figure 18 - SMS Alert Application State Transition Diagram

Detailed requirements and specifications are as follows:

- Web pages will be written in XHTML 1.0.
- Java Servlets will handle web input from the user.
- The Apache Tomcat 5.5 web server will store the Servlet applications.
- When the user requests the web page for the parking garage, the servlet will read from the garage log file the current capacity of the garage and generate an XHTML file which will then display this information to the user.
- A separate XHTML page will enable the user to sign up to a SMS text messaging service which will alert the user via cell phone the current status of the garage. This page will allow the user to specify their name, number, carrier, and time they want the text message to be sent.
- Once the user submits this information, it will be added to a database.
- A Java program will monitor the system time and check the database every minute. If a user wants a text to be sent at the current time, the java program will create an email message to be sent to the user's telephone number (See Figure 19~~Figure 19Figure 19~~).

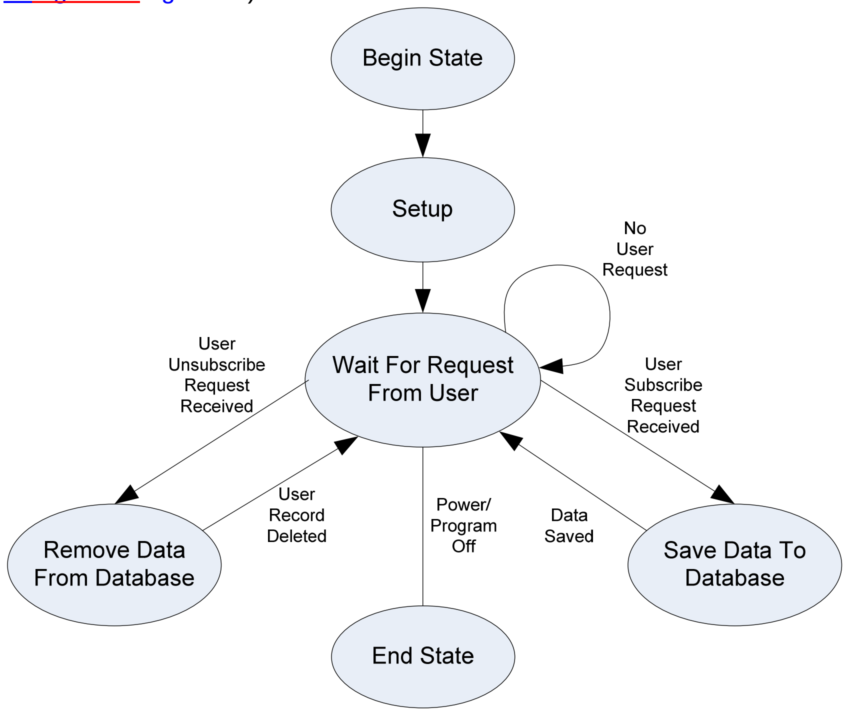

Begin State
Setup
No User Request
Wait For Request From User
User Unsubscribe Request Received
User Subscribe Request Received
Remove Data From Database
User Record Deleted
Power/ Program Off
Data Saved
Save Data To Database
End State

Figure 19 - Servlet State Transition Diagram

III. References

[1] Susan Alderman. (2009, September) Indiana University-Purdue University Fort Wayne. [Online]. http://www.ipfw.edu/news/archives/2009/Sept/14-enrollmentRecords.shtml

[2] Jacob Pitcher. (2009, Spring) Survey of Active Spring 2009 Students. Purdue Qualtrics Survey.

[3] Michael Devault. (2009, Sep.) DevaSys Embedded Systems. [Online]. http://www.devasys.com/usbi2cio.htm

[4] Microsoft. (2009, Sep.) Windows Server 2008 R2 System Requirements. [Online]. http://www.microsoft.com/windowsserver2008/en/us/system-requirements.aspx

[5] Banner Engineering Inc. (2009, Sep.) M-GAGE Q7M Flat-Pak. [Online]. http://info.bannersalesforce.com/xpedio/groups/public/documents/literature/117172.pdf

[6] Banner Engineering. (2009, Sep.) Models PS24-1.andPS115-1.Sensor Interface Modules. [Online]. http://info.bannersalesforce.com/xpedio/groups/public/documents/literature/123566.pdf

[7] Qualtek Electronics Corp. (2009, Sep.) Qualtekusa.com. [Online]. http://www.qualtekusa.com/Catalog/EMI_Filters/pdf/86206001.pdf

[8] IndySpeedway.com. (2009, Sep.) Formula One vs Indy Car. [Online]. http://www.indymotorspeedway.com/vsseries.htm

[9] ABB Electronics Corporation. (2009, Sep.) R600 optocoupler modules. [Online]. http://www.abb-control.com/electronicscat/AC02020.19-22.pdf

[10] Weidmuller. (2009, Sep.) Product catalogue. [Online]. http://catalog.weidmueller.com/procat/Product.jsp;jsessionid=8E30EB96587EA790A4F2FBEF92631B36?productId=([8398940000])&page=Product